University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming deadlines:
  - Homework 8 due on 3/21
  - Lab 8 due on 3/25
  - Homework 9 due on 3/28
  - Assignment 2 due on 3/28

Previous lecture …

- Minimum Spanning Tree Problem

CourseMIRROR Reflections (most confusing)

- how to assign the value for each edge
- The process for finding an articulation point algorithmically was most confusing
- Why low is the minimum of num(v), num(w), lowest low(w)
- The method of visiting every node of a weighted graph using the path with the lightest weight was most confusing.

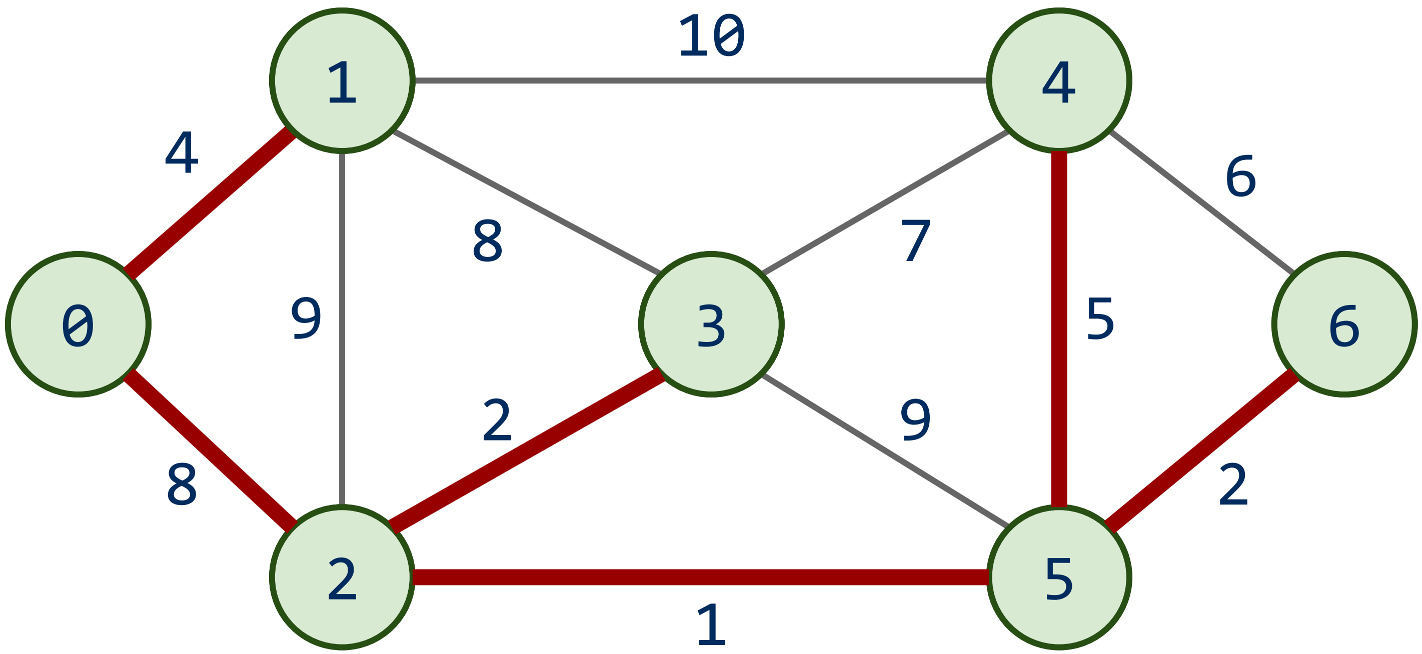
CourseMIRROR Reflections (most interesting)

- Tracing through Depth First and Breadth First Searches
- Low value tracking
- Algorithm for finding articulation points
- The idea of a weighted graph was most interesting.
- I found it interesting how back edges helped with finding articulation nodes
- The simplicity and correctness of Prim’s algorithm
- The articulation point algorithm and minimizing costs with connections

Problem of the Day

- **Neighborhood connectivity project**
  - We want to keep a set of neighborhoods connected with the minimum cost possible
- **Input:** A set of neighborhoods and a file with the following format:
  - neighborhood i, neighborhood j, cost of connecting the two neighborhoods
  - …
- **Output:** A set of neighborhood pairs to be connected and a total cost
  - We can go from any neighborhood to any other **(connected)**
  - The total cost should be minimum (i.e., as small as it can be) **(minimal cost)**

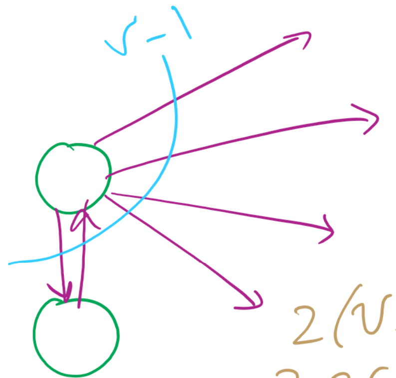
Prim’s algorithm



Runtime of Prim's

- At each step, check all possible edges
- For a complete graph:
  - First iteration:
    - v - 1 possible edges
  - Next iteration:
    - 2(v - 2) possibilities
      - Each vertex in T shared v-1 edges with other vertices, but the edges they shared with each other already in T
  - Next:
    - 3(v - 3) possibilities
  - …
- Runtime:
  - $\Sigma_{i\ =\ 1\ \text{to}\ v}\ (i * (v - i))$
    - Evaluates to $\Theta(v^3)$

Analysis of Naïve implementation of Prim's MST Algorithm



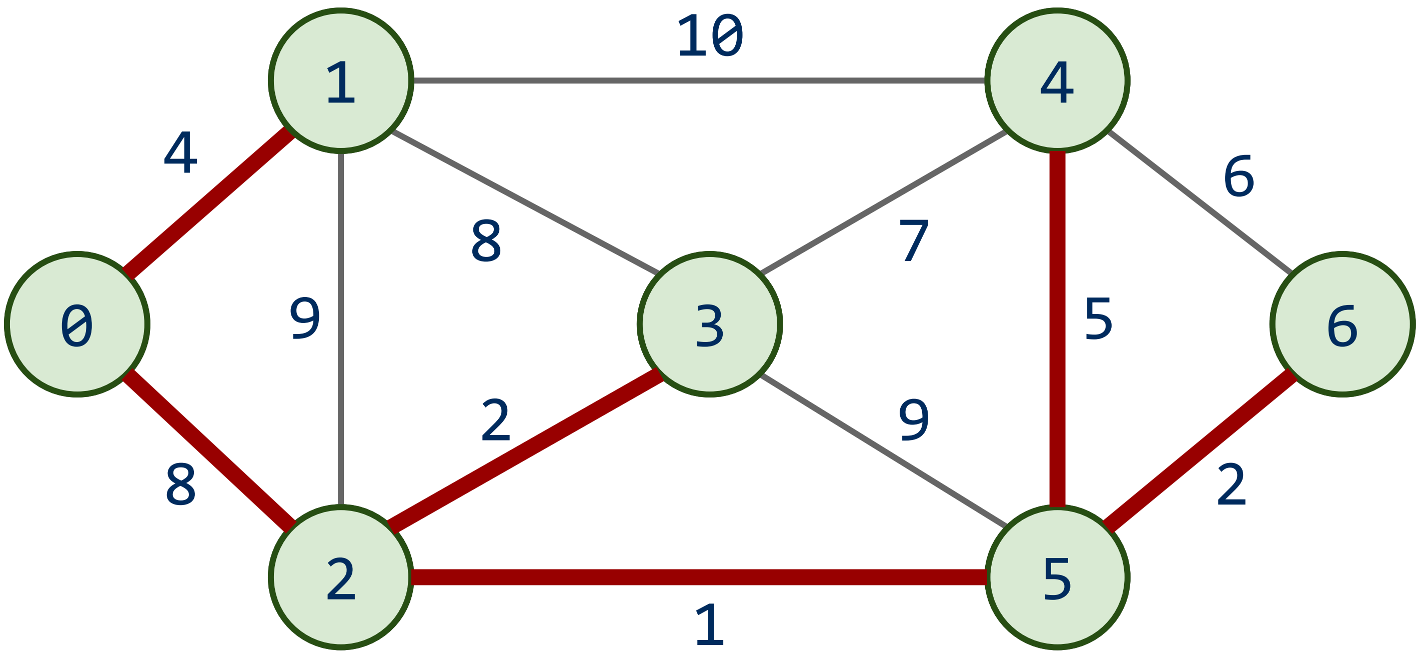
$$2(v-1)-2$$
$$= 2v-4 = 2(v-2)$$

$$(v-1) + 2(v-2) + 3(v-3) + \cdots$$
$$= \sum_{i=1}^{v} i(v-i)$$
$$= \Theta(\#\text{terms} * \text{largest term})$$
$$= \Theta\left(v * \frac{v}{2}\left(v - \frac{v}{2}\right)\right)$$
$$= \Theta(v * cv^2) = \Theta(v^3)$$

Do we need to look through all remaining edges?

- No! We only need to consider the *best* edge possible for each vertex!

Prim's algorithm



	0	1	2	3	4	5	6
Parent:	--	0	0	2	5	2	5
Best Edge:	0	4	8	2	5	1	2

OK, so what's our runtime?

- For every vertex we add to T, we'll need to check all of its neighbors to check for edges to add to T next
  - Let's assume we use an adjacency matrix:
    - Takes $\Theta(v)$ to check the neighbors of a given vertex
    - Time to update parent/best edge arrays?
    - Time to pick next vertex?
  - What about with an adjacency list?

Prim's MST Algorithm

v

Seen: 0

Parent: -1

Best Edge: ∞

BestEdge[start] = 0

for i = 0 → v-1

  find an unseen vertex w with the minimum BestEdge value ← $\Theta(v)$

  Seen[w] = 1

  for each neighbor x of w ← $\Theta(v)$

    if (BestEdge[x] > edgeweight of (w, x))

      BestEdge[x] = edgeweight of (w, x)

      Parent[x] = w

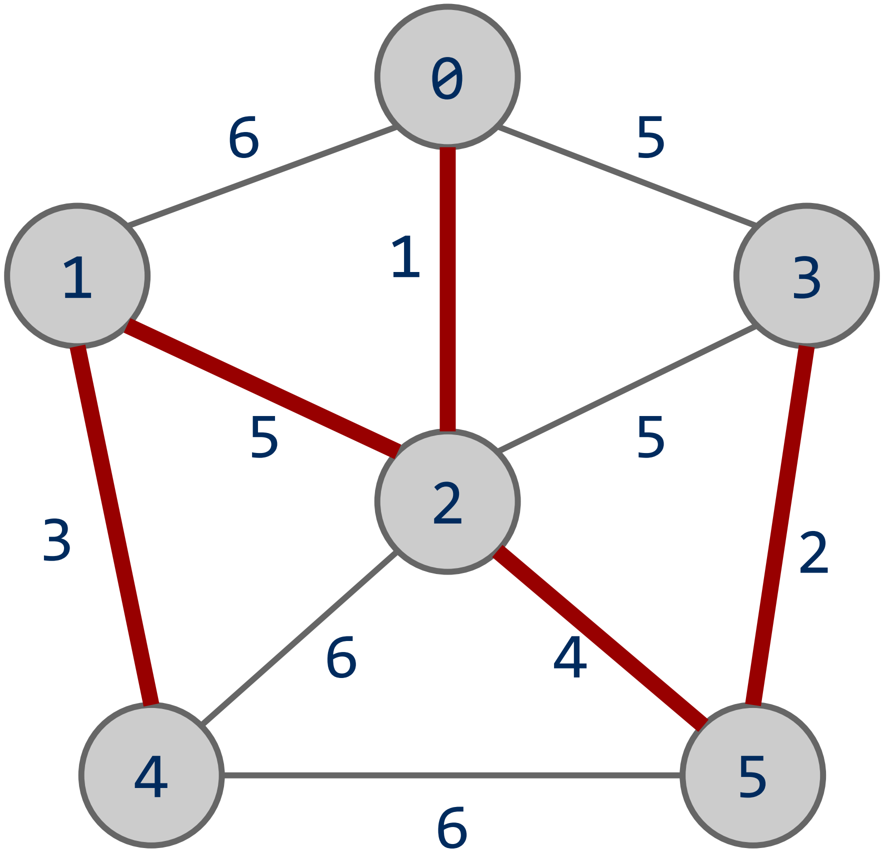
$\Theta(1)$

$\Theta(v^2)$

Another MST algorithm

- Kruskal's MST:
  - Insert all edges into a PQ
  - Grab the min edge from the PQ that does not create a cycle in the MST
  - Remove it from the PQ and add it to the MST

Kruskal's example



PQ:

1: (0, 2)
2: (3, 5)
3: (1, 4)
4: (2, 5)
5: (2, 3)
5: (0, 3)
5: (1, 2)
6: (0, 1)
6: (2, 4)
6: (4, 5)

Kruskal's runtime

- Instead of building up the MST starting from a single vertex, we build it up using edges all over the graph
- How do we efficiently implement cycle detection?

Kruskal's Runtime: Take 1

e iterations

remove $\log e$

cycle detection $\Theta(v+e)$

DFS/BFS

$e(v+e) = \Theta(e^2)$

Please submit your reflections by using the CourseMIRROR App

If you are having a problem with CourseMIRROR, please send an email to coursemirror.development@gmail.com